Analysis on Romance Movies with the IMDb Dataset

1. Introduction

In this project, we use the IMDb datasets which consist of information about video resources across the world that are documented by the IMDb website. Our project mainly focuses on analyzing the past, the present and the future of romance movies, trying to derive some internal pattern of the romance movie industry.

There are three parts in our project. First, we complete a movie genre trend analysis in general. We aim to visualize and analyze movie trend in the past century, categorized by genres. We start with romance movies, and later expand the scope to four genres: romance, action, science fiction and horror. We mainly focus on how people's taste of movie genres has changed over time in this section.

Second, we complete a romance movie export analysis. We aim to understand which countries export most romance movies and to which countries are exported. The objection is to visualize with an interactive map which could indicate the number of exports and relationships between countries in terms of romance movies.

Finally, we utilize machine learning to find an optimal classifier that classifies romance movies into two categories: recommended and unrecommended. After training, the classifier can tell the user whether a romance movie deserves watching or not given some basic information of the movie as input.

(Please note that the 'title.basics.tsv.gz' and other IMDb datasets are not included in our zipfile. Please find them on the IMDb dataset website at https://datasets.imdbws.com/)

2. Part I: Movie Genre Trend Analysis

In this part, we mainly aim to visualize and analyze movie trend in the past century, categorized by genres. We start with romance movies, and later expand the scope to four genres: romance, action, science fiction and horror. We mainly focus on how people's taste of movie genres changes over time in this section.

2.1. Data Preprocessing

We take two files 'title.basics.tsv.gz' and 'title.ratings.tsv.gz' from IMDb website as our raw dataset in this part, read in as two Pandas dataframes: basics and ratings. We mainly used these attributes for each title: tconst, titleType, genres, and startYear from 'title.basics.tsv.gz'; tconst, averageRating and numVotes from 'title.ratings.tsv.gz'.

First, we use titleType to find all movies in the dataset, and merge ratings with it on tconst, which the unique identifier for each movie. We define a new attribute Popularity as follows: for each movie m,

$$\text{Popularity}(m) = \text{average rating}(m) * \text{number of votes}(m) \qquad (1)$$

After we use (1) to compute the acceptance for each movie, we drop all columns except tconst, genres, startYear, and Popularity. Here, we reduce the dataset from the original 5687200 x 9 matrix of basics and 914862 x 3 matrix of ratings to a 450481 x 4 matrix. Finally, for each movie genre G we aim to study, we find and save movies of G in its separate dataframe by parsing and checking the genres string of all rows.

2.2. Time Series Visualization

2.2.1 Measurement Method

In this part, our goal is to visualize how IMDb users' preference on movies from different genres changes over time, sampled by year. We have two measurements for genre popularity:

$$\text{Count}(G, Y) = \Sigma m \; 1 \; (\text{genre\_m} = G, \text{year\_m} = Y) \qquad m \in \{\text{all movies}\} \qquad (2)$$

$$\text{Popularity}(G,Y) = \Sigma m \; \text{Popularity\_m} \; (\text{genre\_m} = G, \text{year\_m} = Y) \qquad m \in \{\text{all movies}\} \qquad (3)$$

Formula (2) computes the total number of movies produced in a genre per year, and formula (3) computes total Popularity of a genre per year, in which Popularity is the attribute we define in (1).

2.2.2 Aggregation into Time Series

For every genre G, we aggregate its corresponding dataframe twice grouping by startYear. For the first time, we count its movie production per year, and for the second time we sum up popularity for every year. With four targeted genres, we get eight time series data sets. We aggregate one more time for the total count of movies per year:

$$\text{Count}(Y) = \Sigma m \; 1 \; (\text{year\_m} = Y) \quad m \in \{\text{all movies}\} \qquad (4)$$

We plot the results in following graphs:

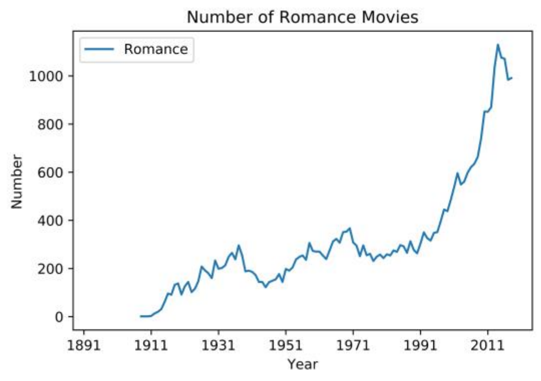


Figure (1)

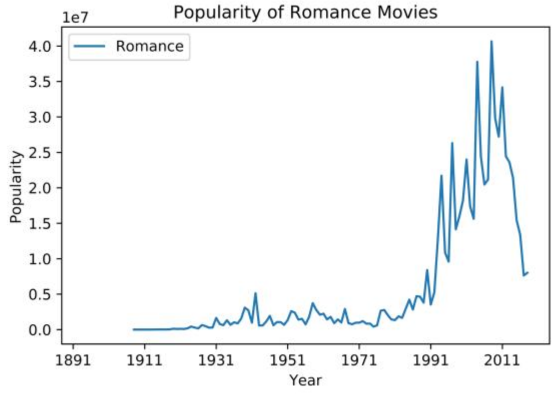


Figure (2)

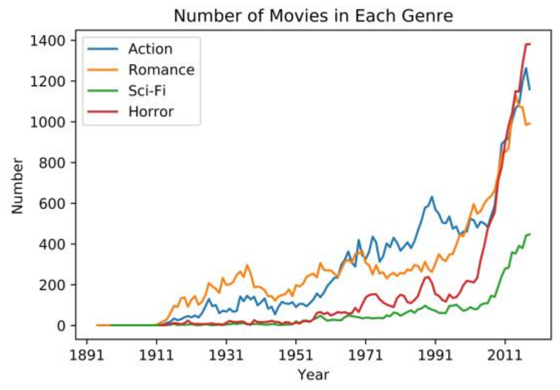


Figure (3)

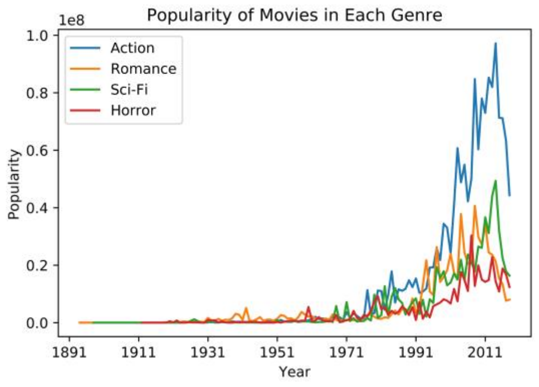


Figure (4)

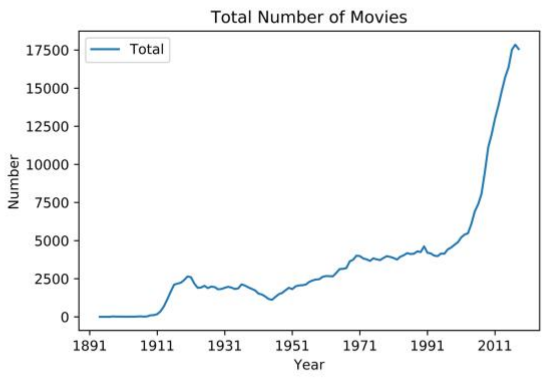


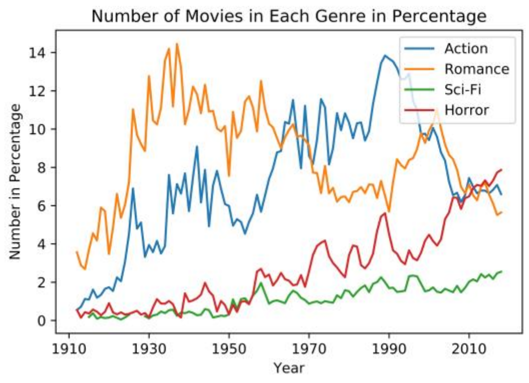
Figure (5)

2.2.3 Industry Fraction Regularization

From Fig (3) we notice that all the count for all four movies genres have the general tendency of growing, and the relative difference between genres is not obvious. We know from Fig(5) that the movie industry has been growing continuously over the years, so we cannot conclude that people are more interested in those genres than before. We introduce a new measurement Industry Fraction(IF) as follows:

$$IF(G, Y) = Count(G,Y) / Count(Y)$$
$$= \Sigma m\ 1\ (genre\_m = G, year\_m = Y) / \Sigma m\ 1\ (year\_m = Y)$$
$$m \in \{all\ movies\}$$

We compute industry fraction of each genre of each year for all four genres, and plot the results as follows:

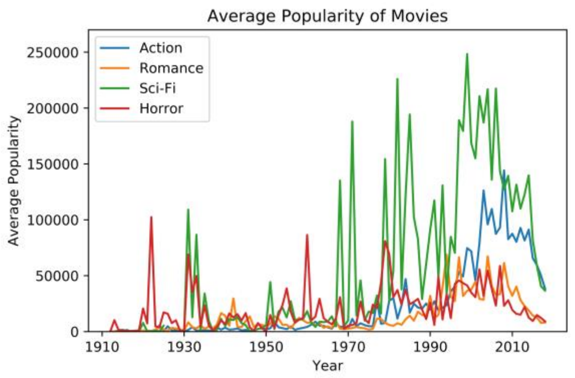


2.2.4 Average Popularity Regularization

Fig (4) shows that Popularity(G, Y) is similar to Count(G,Y), but not exactly identical. For example, we can see that science fiction movies are produced much less than the other three, but it exceeds horror movies and romance movies in popularity in recent decades. To better display how welcomed movies from a genre are, we add one more regularization on our time series, the Average Popularity(AP) of movies from a genre per year:

$$AP(G, Y) = Popularity(G,Y) / Count(G,Y)$$
$$= \Sigma m\ Popularity\_m\ (genre\_m = G, year\_m = Y) / \Sigma m\ 1\ (genre\_m = G, year\_m = Y)$$
$$m \in \{all\ movies\}$$

We compute average popularity of each genre of each year for all four genres, and plot the results as follows:



2.3. Discussions and Future Works

From the above Figures, we can spot some interesting facts about IMDb users' preference of movie genres. For example, romance movie was more popular back in the old days from 30s to 60s; romance movies and action movies went through dramatic decrease of their share in the industry in the recent ten years; science fiction movies are produced in the least amount but are highly welcomed and rated by people.

Although in this part of our project, we see neither obvious cyclic pattern of the same genre over time, nor obvious similar pattern of different genres. In the future, we can try different time series transformation matrices, such as discrete Fourier transformation(DFT), and time series analysis algorithms to mine potential underlying patterns and achieve time series forecasting on movie genre popularity.

3. Part II: Romance Movie Exports Analysis

In this part, our goal is to understand which countries export most romance movies and to which countries are exported. The objection is to visualize with an interactive map which could indicate the number of exports and relationships between countries in terms of romance movies. To do so, we mainly utilized 3 datasets in this case and imported several packages, such as pandas, plotly and numpy. We take three steps in this analysis: data cleaning, data processing and visualizing.

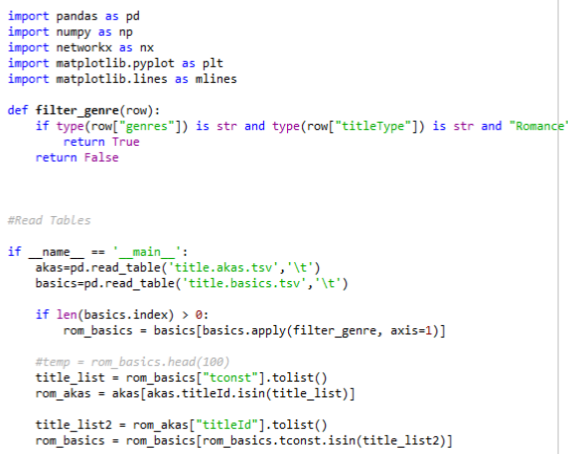
3.1 Data Cleaning

In this part, our goal is to derive a list of romance movies. There are three main datasets being utilized: 'Country_loc.tsv', 'title.akas.tsv', and 'title.basics.tsv'.

- *Country_loc.tsv:*
  It includes the country name (str), Alpha-2 code (str), latitude (average) (float) and longitude (average) (float). This dataset mainly serves to provide geographic information to help plot our findings on a world map.
- *title.akas.tsv:*
  This dataset includes the titleId (str), title (str), titleType (str) region (str), and isOriginalTitle (boolean). 'titleId' would be used as a reference connecting 'title.akas.tsv' and 'title.basics.tsv'. 'Title' is the name of the movie in different languages, and 'region' explains which region is that language spoken. 'titleType' indicates the format of each film. In our case, we want to use this column to filter out those non-movie films. 'isOriginalTitle' indicates if the title of the movie is

the original one. This dataset can help us understand to which countries each movie was exported to.

- *'title.basics.tsv':*
  This dataset includes the tconst (str) and genres (str). 'tconst' is the identifier of each movie that matches the 'titleId' from title.akas.tsv. 'genres' introduces the genres of each movie, stored in an array. We utilize 'genres' to sort out only romance movies.



```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.lines as mlines

def filter_genre(row):
    if type(row["genres"]) is str and type(row["titleType"]) is str and "Romance" in row["genres"].split(",") and row["titleType"]=="movie":
        return True
    return False


#Read Tables

if __name__ == '__main__':
    akas=pd.read_table('title.akas.tsv','\t')
    basics=pd.read_table('title.basics.tsv','\t')

    if len(basics.index) > 0:
        rom_basics = basics[basics.apply(filter_genre, axis=1)]

    #temp = rom_basics.head(100)
    title_list = rom_basics["tconst"].tolist()
    rom_akas = akas[akas.titleId.isin(title_list)]

    title_list2 = rom_akas["titleId"].tolist()
    rom_basics = rom_basics[rom_basics.tconst.isin(title_list2)]
```

We derived a list of romance movies. We set up a helper function named as 'filter_genre(row)' to help filter out romance movies. We read in 'title.akas.tsv' and 'title.basics.tsv' as pandas dataframe akas and basics. Applying filter_genre to akas resulted in a new dataframe 'row_basics', and then I used the titleId and tconst from 'title.akas.tsv' and 'title.basics.tsv' to double check if movies, after cleaning, exist in both datasets.

3.2 Data Processing

In this part, our focus is to build 2 pandas dataframe for visualization: one (country_loc_refine) is to indicate the countries, number of exports, longitudes and latitudes for plotting dots with plotly, and the other one (for_plotly) is to indicate the number of exports among countries for plotting edges with plotly.

Firstly, we created a new panda dataframe named as for_visual_movie to summarize the key information of each movie before we create the two dataframes for visualization. It includes columns: 'name', 'Original_Region', 'No_Exports', and 'Export'. 'name' indicates the title of the movie, 'Original_Region' illustrates the exporting country, and 'No_Exports' and 'Export' are referring to the number and the list of exported countries respectively. We employed a helper function to fill in for_visual_movie. 'reorganize(name, title_id, rom_akas, for_visual)' is used to identify the original country of each movie in 'title.akas.tsv', count the number of countries being exported to and generate a list of those exported countries. This function returns an updated for_visual_movie with a new row being added.

After finishing with for_visual_movie, we prepared the country_loc_refine. This dataframe is used to plot the dots on the world map with plotly. Thus, it includes columns 'Abb', 'Name', 'Lat', 'Lon' and 'No_Exports'. 'Abb' and 'Name' indicate the Alpha-2-code and name of each exporting country respectively. 'Lat' and 'Lon' stand for the latitude and longitude of each country, and they are referenced from the initial dataset Country_loc.tsv. 'No_Exports' is the total number of exports, and it is the sum of No_Exports of movies from the same original country in for_visual_movie.

Last but not least, we prepared the dataframe, for_plotly, for plotting the edges with plotly. for_plotly includes columns of 'Origin', 'Dest', 'Number of movies', 'OriLat', 'OriLon', 'DestLat', and

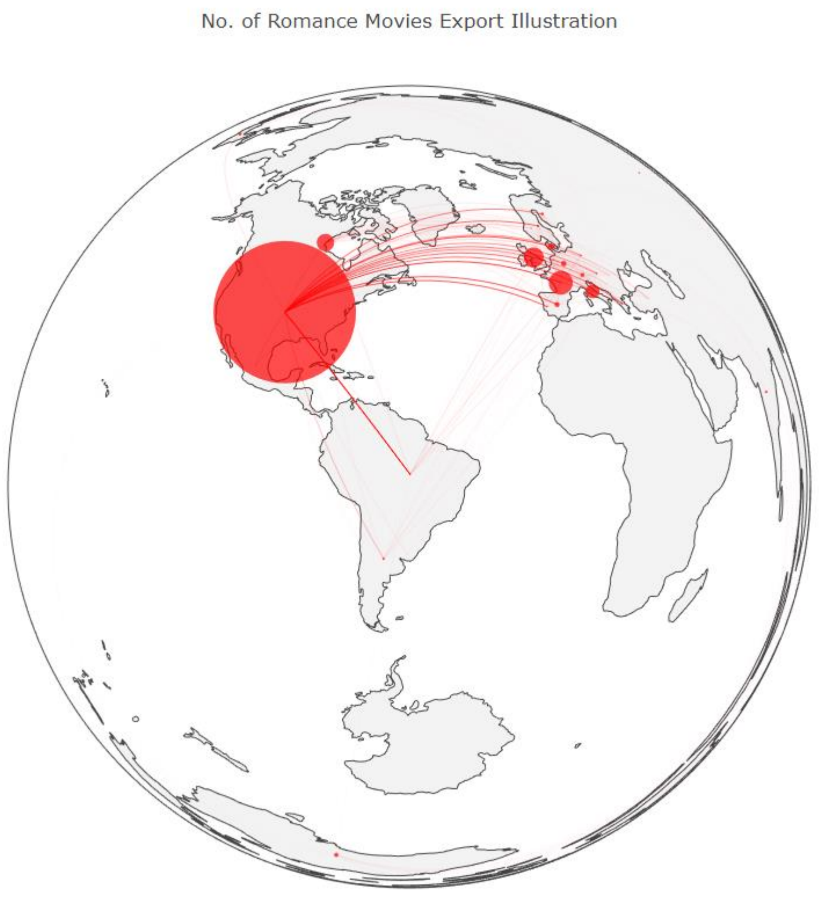
'DestLon'. The 'Origin', 'Dest', 'OriLat', 'OriLon', 'DestLat', and 'DestLon' represent the original and destination countries names, and their longitudes and latitudes information. 'Number of movies' refers to the total number of movies being exported from one country to another. In order to get this data, we created a second function is 'count_number(for_visual_movie, original, dest)' which iterates through the for_visual_movie to count the total number of romance movies being exported from original to destination country.

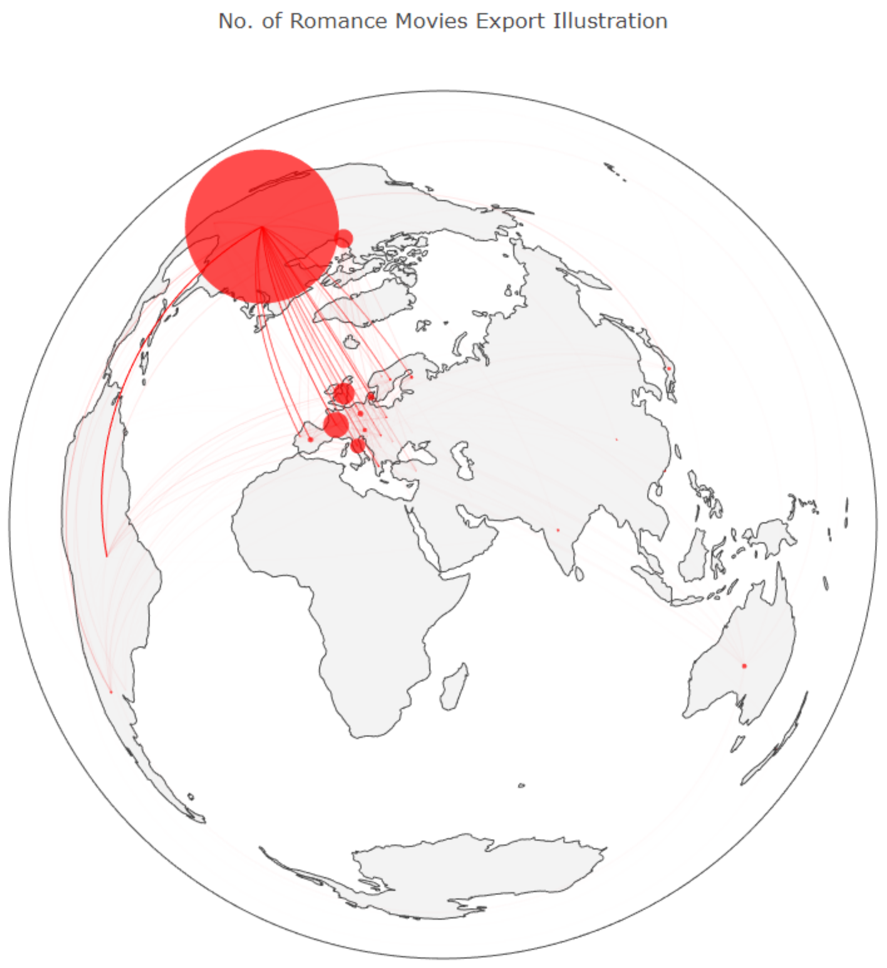
After preparing country_loc_refine and for_plotly, we exported them to a csv format for further visualization.

3.3 Visualization (The source code for this part is kept in separate .py file named as 'Part II Visualization II.py')

For visualization, we used the plotly package which could help generate an interactive map. With plotly, we aim at illustrating the number of exports of each country and the number of movies being exchanged in between every 2 countries. We mainly utilized country_loc_refine as the data reference for plotting the dots. The size of the dots are indicators for the number of exports from every country. Other than that, for_plotly are utilized to illustrate the edges between dots (countries). The opacity of the edges is determined by the ratio 'Number of Movies' by max('Number of Movies').

The output is presented below.





When choosing the packages for visualization, we decided to adopt plotly other than NetworkX. It is not only because plotly is more interactive, but also the incompatibility between Anaconda and BaseMap made it unreasonably challenging to choose NetworkX.

3.4 Analysis

Here we have 3 important observations from this interactive map. The first one is that the highest exporting countries in terms of volume is U.S., followed by France, U.K., Canada and Italy. Secondly, most of exports happen in the western world, while a minimal amount of exports happened in the rest of the world. Lastly, there is no significant edge between countries in western world and those in the east.

From those key observations, my interpretation is that the concept of “Romance and Love” has higher cultural uniformity in the western world, while numerous cultural implications of “love” in the East might result in few exports of their romance movies. Such interpretation could be used to explain why the correlations among western countries are more significant than those among countries in the rest of the world.

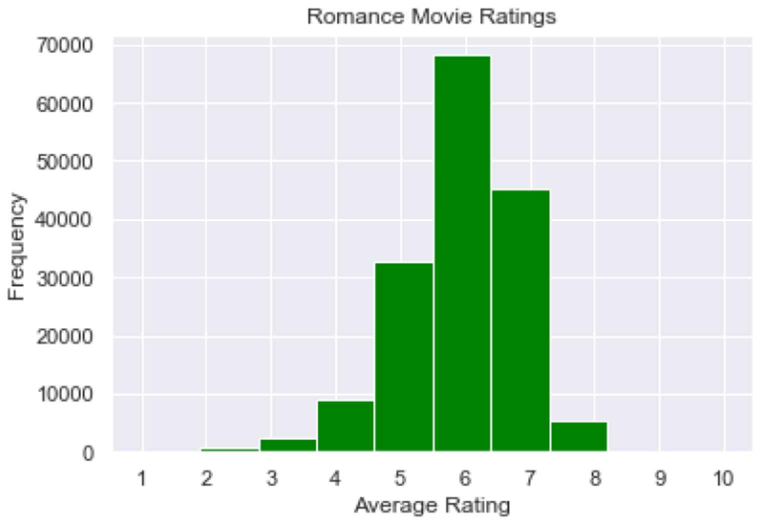
Apart from that, based on our data, it also shows that the number of movies produced in the western countries is highly likely to be higher than that of the rest of the world. It could be attributed to the expertise and capabilities of producing movies in general, or it could be attributed to the differences in preferences for genres in different parts of the world.

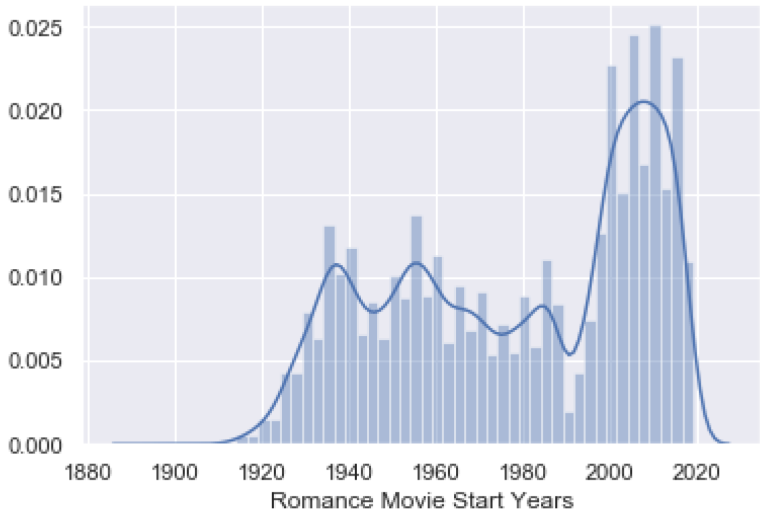
Finally, we believe that there is a gap between eastern and western understanding of romance movies. The weak correlation between countries in western world and those in the east illustrates that people in those different parts of the world do not find movies from each other much attractive. It could be because the filming skills, story-telling methods and perception of being romantic do not match in the east and the west.

4. Part III: Binary Classification of Romance Movie Quality

4.1 Data Preprocessing

Except for the data cleaning process illustrated in the previous section, we also center the numerical variable "year" around 1980 by deducting 1980 from every entry. This might help reduce computation difficulty in the later process and prevent matrix entries from being too large. Moreover, the rating 6.5 is chosen as the threshold because nearly 80% of the films are below this score. Thus, we find it to be a reasonable score that distinguishes good movies from bad ones from the viewer's perspective.





4.2 Get feature Matrix and Label Array

To make the training and testing process efficient, we use a sample size of 10000. Among the four features we choose, the start year is a numerical variable, so we just use integers to denote the value. For the other categorical variables, we decide to use one-hot encoding. For each variable, there are two steps in the encoding process:

1) Extract a dictionary that maps each category with a unique index number;
2) If the data sample belongs to the category, put a 1 at the unique index and 0's at all the other places.

All of the above is implemented using simple for-loop. Then, concatenate the one-hot vectors together along with the number denoting the start year. Hence, we get the feature matrix.

To obtain the label array, we simply use another mask that filters out the movie entries with rating higher than 6.5.

4.3 Dimensionality Reduction

At first, we directly feed the feature matrix to the grid search svm. However, it takes such a long time for the computer to finish fitting that we decide to analyze the matrix before doing any computation. We find that the matrix has size 10000 * 497 and is very sparse because it is obtained by one hot encoding. Thus, it seems that though large, the matrix itself does not contain that much useful information. That is why we decide to do dimensionality reduction before training.

The method we use is sklearn.TruncatedSVD. This function set allows us to create a truncator with a user defined parameter k and transform any n*m matrix into a n*k one. While drastically reducing the dimensionality of the input matrix, TSVD finds the feature correlation of the input and thus preserves important information about the pattern of the features.

Both NMF, PCA and SVD are decomposition techniques to be used on sparse matrix. However, we chose SVD because of the following reasons.

1) SVD factors can contain both positive and negative entries, but NMF factors are strictly positive. Since we have negative values in our matrix, we chose SVD over NMF. However, in some settings such as text mining, if all entries of original matrix are positive and have physical meanings like term frequencies, one would want the factors to be positive so that physical connections are be directly made. In this case, NMF is better.
2) 2. There is a way to do an SVD on a sparse matrix that treats missing features as missing (using gradient search). But for PCA, you can only treat missing features as zero.

4.4 Train Classifiers

In this section, we use a training size of 8500 and a test size of 1500.

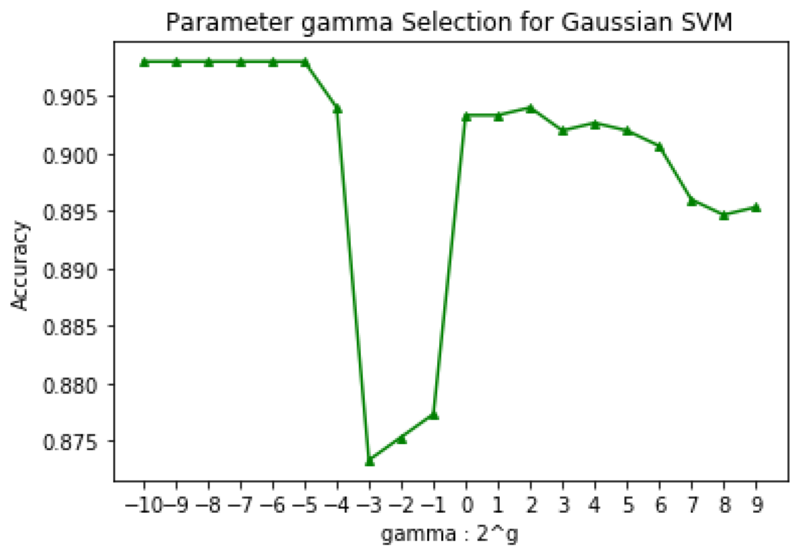
4.4.1 Dummy classifier

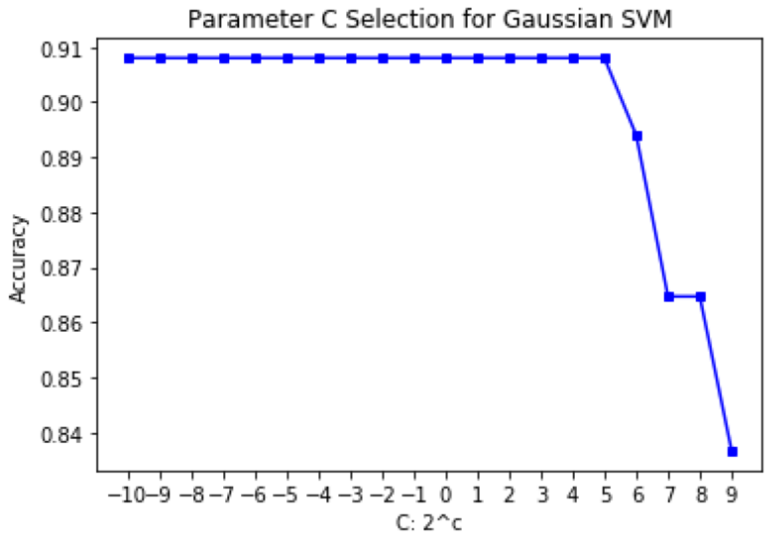
First, to set a baseline for the accuracy, we train a dummy classifier using sklearn.DummyClassifier. The score turns out to be around 0.5, which is just like randomly giving the sample a label.

4.4.2 SVM with gaussian kernel

First, to find an optimal set of parameter, we use sklearn.StratifiedKFold and sklearn.GridSearchCV. These two methods allow us to do a cross validation search over the range of parameters defined by ourselves. The grid search is an especially useful technique because it builds a model for every combination of hyperparameters specified and evaluates each model. Hence, we don't have to write for loops ourselves.

The result is plotted in figure below. In our case, the best C parameter is 1 and the best gamma is 0.01. The C parameter balances between correctly classifying training instances and maximizing decision function's margin, while γ controls how far the influence of a single training example can reach. Low values of gamma mean that the influence can be far and high values mean close.



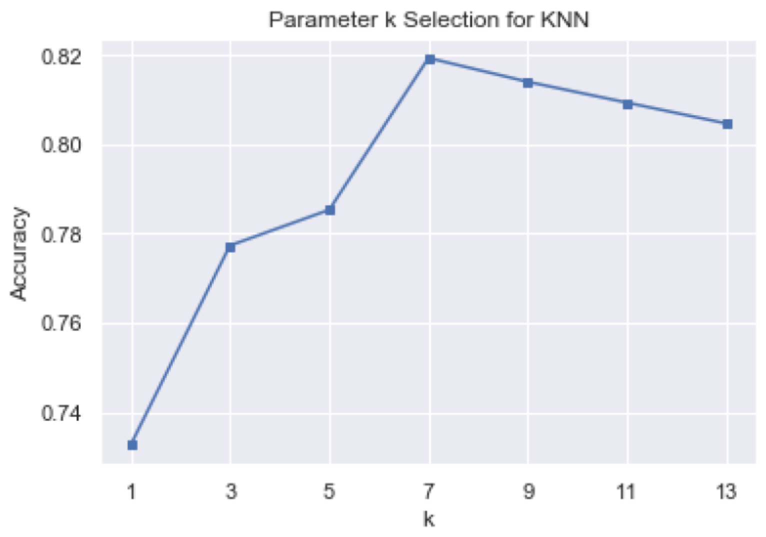


4.4.3 Bernoulli Naive Bayes

This classifier is chosen to illustrate that not all classifiers can be used to train our data. Remember that our input is obtained by one-hot encoding, so there is a natural dependence between columns of the feature vectors. Typically, a 1 in one column means that it is less likely to have a 1 in the next column since the category is unique. However, naive Bayes classifier is one of the simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. The independence assumption seems to not fit our dataset. Hence, we can see from the test score that it does not do a good job.

4.4.4 KNN

Finally, we implement a KNN using the sklearn.neighbors.KNeighborsClassifer(). To tune the hyperparameter k. We write a for loop that iterates over a specified range and get the following curve. We can see that small k tends to underfit while large k tends to overfit.



4.5 Findings

We choose the four models above because they are common binary classifiers that predict a label given a list of input features. Normally, these methods are sufficient to deal with simple datasets. However, to get a stronger classifier, we can further combine individual models using adaboost or bagging to get a more accurate result. But for the IMDb dataset, these methods are sufficient, and we can explore some properties of each model during the training process.

	Features	Accuracy
Dummy Classifier	uniform	0.5193
Gaussian SVM	sklearn.StratifiedKFold sklearn.GridSearchCV	0.908
Bernoulli Naïve Bayes		0.746
KNN	uniform weights/distance best k: 7 train score: 0.8267	0.8193

The accuracy scores of the four models are given in the table above. We can see that in general, the Gaussian SVM performs the best and the BNB has the lowest test score.

Moreover, there are several other findings that we want to emphasize here.

1) While implementing the classifiers and analyzing the feature matrix, we find that the features are not absolutely deterministic in terms of the label we defined. In other words, there might not be such a strong relationship between the input and output. Hence, the training result is not as satisfiable as we thought. However, we presume that the score can be higher if we have more training samples, but due to the limited CPU on our personal laptop, we cannot test this assumption.
2) At present, our models are still too simple to deduce the complicated pattern of the dataset. In the future, we might want to neural networks to further strengthen our project. Besides, we can apply further techniques in reducing noises in preparation for the training.
3) At least in our project, we find that tuning hyperparameters is very important. The set of parameters you chose can greatly affect the final score you have.

5. Conclusion

In conclusion, our project analyzes the past and present of the romance movie industry. Our observations can be summarized as below.

1. As the film industry developed with time, romance movies experienced a fast development.
2. Due to political, economic and cultural reasons, different countries understand romance differently and have distinct preferences.
3. People have different standard/potential bias towards the quality of movies. It is hard to categorize a movie as good or bad.

We think that this project is not only interesting but also allows us to learn a lot. On one hand, we strengthened our skills in using python to do data mining and machine learning. On the other hand, we learned how to corporate with each other and at the end of this quarter, we are so glad that we have known each other and finished a project together.